The opinion in support of the decision being entered today is *not* binding precedent of the Board.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte DAVID K. POULSEN,
SANJIV M. SHAH, PAUL M. PETERSEN,
GRANT E. HAAB, AND JAY P. HOEFLINGER

Appeal 2007-1959
Application 10/039,789
Technology Center 2100

Decided: September 11, 2007

Before ANITA PELLMAN GROSS, MAHSHID D. SAADAT,
and ST. JOHN COURTENAY, III, *Administrative Patent Judges.*

SAADAT, *Administrative Patent Judge.*

DECISION ON APPEAL

STATEMENT OF THE CASE

Appellants appeal under 35 U.S.C. § 134(a) from a Final Rejection of claims 1, 3, 6-8, 10-15, 18, and 20-26, which are all of the claims pending in this application, as claims 2, 4, 5, 9, 16, 17, and 19 have been canceled. We have jurisdiction under 35 U.S.C. § 6(b).

Appellants have invented a method and an apparatus for efficiently and cost effectively implementing reduction operations over multiple computer platforms (Specification 3).

Claim 1, which is representative of the claims on appeal, reads as follows:

1. A method comprising:

receiving a first program unit in a parallel computing environment, the first program unit including a reduction operation associated with a set of variables;

translating the first program unit into a second program unit, the second program unit including a set of one or more instructions to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit; and

translating the first program unit into the third program unit, the third program unit including a set of one or more instructions that encapsulate the reduction operation to perform an algebraic operation on the variables.

The prior art relied upon by the Examiner in rejecting the claims on appeal is:

Poulsen	US 5,812,852	Sep. 22, 1998
Sundaresan	US 5,937,194	Aug. 10, 1999
Hardwick	US 6,212,617 B1	Apr. 3, 2001

The Examiner rejected claims 1, 3, 6-8, 10-15, 18, 20, 21, 23, and 26 under 35 U.S.C. § 103(a) as being unpatentable over the teachings of Poulsen and Sundaresan, and claims 22, 24, and 25 over the teachings of Poulsen, Sundaresan, and Hardwick.

ISSUE

The issue is whether Appellants have shown that the Examiner erred in rejecting the claims under 35 U.S.C. § 103. Appellants urge that the translation of a parallel computer program in Poulsen is a single translation into a second parallel program and does not include a third program based on another translation of the first program (Br. 12). Based on such interpretation, Appellants argue that one of ordinary skill in the art would not combine Poulsen with Sundaresan since Sundaresan does not indicate that such translation is needed (Br. 13). Therefore, the issue turns on whether there is a legally sufficient justification for combining the disclosures of Poulsen and Sundaresan and, if so, whether the combination of the applied references teaches the claimed subject matter, including translating a first program unit into a second program unit and into a third program unit, with each program unit performing its corresponding function.

FINDINGS OF FACT

The following findings of fact (FF) are relevant to the issue involved in the appeal and are believed to be supported by a preponderance of the evidence.

1. Appellants describe the claimed term "program unit" as "a collection of statements in a programming language that may be processed by a compiler or translator" (Specification 7:13-15).

2. Poulsen relates to privatizing global storage objects in parallel computer programs to provide per-thread private copies of these global objects when executing parallel computer programs (col. 1, ll. 28-33).

3. Poulsen defines privatization as the process of partitioning storage objects in a global memory address space among multiple processors or threads of execution. Whereas a global storage object, in a global memory address space, is accessible to multiple processors or threads, an access to a privatized global storage object, by a particular processor or thread, results in accessing one of multiple, local copies of the global object (col. 1, ll. 37-44). The parallelism-enhancing effects of privatization include eliminating memory access conflicts (formally referred to as data dependences), partitioning storage and thereby partitioning parallel tasks, and increasing memory reference locality (col. 1, ll. 45-49).

4. Poulsen further discloses a three-step program transformation which includes declaring a compound object (e.g., a structure) that encapsulates the object to be privatized, translating the references to the object to be privatized references to the new compound object, and inserting a library call that, when executed, instantiates and initializes the appropriate thread-private memory for the compound object and initializes a pointer variable to point to this thread-private memory (col. 5, ll. 7-20).

5. Figure 1 of Poulsen shows augmenting program 100 via translation means 120, according to privatization specification 110, to produce a translated parallel computer program 130. Translated program 130 is linked with a runtime support library 140 by a general purpose computer's linker 150 to produce an executable parallel computer program 160, which accesses the global storage objects specified in privatization specification 110 in a privatized manner (col. 8, ll. 27-45).

6. FIG. 2 of Poulsen is a flowchart of the algorithm performed by translation means 120 of Figure 1. The inputs to the algorithm are program 100 and privatization specification 110; the output is translated program 130 (col. 8, ll. 46-49).

7. In Figure 2 of Poulsen, Step 200 determines if there are any more procedures left to be processed in program 100. If not, the algorithm is complete; otherwise, the next procedure is processed beginning at step 210. Step 210 determines if there are any more global storage objects to be privatized in the current procedure. If not, further procedures are considered at step 200; otherwise, the next global storage object in the current procedure is privatized beginning at step 220. Step 220 declares a privatizable storage object (i.e., a structure), which encapsulates the current global storage object, and declares a pointer to that structure (col. 8, ll. 51-61).

8. Step 240 in Figure 2 determines if there are any more parallel regions left to consider in the current procedure. If not, further global storage objects in the current procedure are considered at step 210; otherwise, the next parallel region in the current procedure is processed beginning at step 250 by translating the references to the current global storage object into references to the structure declared in step 220. Step 260 inserts a library call, to initialize library data structures for the current parallel region, at the beginning of the parallel region, while step 270 inserts a library call, to initialize the pointer declared in step 220 to point to the appropriate thread-private memory for the current global storage object, at the beginning of the current parallel region (after the initialization library call inserted in step 260) (col. 8, l. 64 through col. 9, l. 12).

9. Sundaresan relates to a generic reduction object for data parallelism (col. 1, ll. 45-47), wherein a reduction operation is described as one that reduces N values distributed over N tasks and includes operations such as sum and product, maximum and minimum, and logical Boolean operators (col. 1, ll. 52-63).

10. Sundaresan provides for a generic reduction object for data parallelism wherein a data-parallel reduction operation is performed by a group of threads or a rope participating in a multi-level two-phase tree structure (col. 3, ll. 61-67). By separating a reduction object template and type-specific reduction object from the actual reduction operation, the same reduction skeleton object may be used for all reduction operations within a rope, and also a type-specific reduction object, once created, may be reused for different reduction operations of the same type (col. 5, ll. 7-13).

11. Sundaresan further discloses data-parallelism through a reduction operation where each thread contributes a value, and the values are reduced using a function to obtain and return a reduced value to each of the threads (col. 7, ll. 13-16).

12. Hardwick relates to parallel processing methods in which unnecessary inter-process communications are eliminated by using a lazy collection oriented data type (col. 4, ll. 21-25) such as a vector (col. 4, ll. 35-40).

13. Hardwick further discloses that a basic data-parallel data structure is a vector which may be formed from any of the basic C datatypes or user-defined datatypes (col. 6, ll. 30-34). Hardwick further describes "portability" across both shared and distributed memory machines as the

benefit of vector operations (col. 6, ll. 42-45). The data-parallel operations are also taught by Hardwick to be translated into loops over sections of a vector local to each processor (col. 6, ll. 46-49).

PRINCIPLES OF LAW

To reach a conclusion of obviousness under § 103, the Examiner bears the burden of producing factual basis supported by teaching in a prior art reference or shown to be common knowledge of unquestionable demonstration. Our reviewing court requires this evidence in order to establish a prima facie case. *In re Piasecki*, 745 F.2d 1468, 1471-72, 223 USPQ 785, 787-88 (Fed. Cir. 1984).

Furthermore, the test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art. *See In re Kahn*, 441 F.3d 977, 987-88, 78 USPQ2d 1329, 1336 (Fed. Cir. 2006), *In re Young*, 927 F.2d 588, 591, 18 USPQ2d 1089, 1091 (Fed. Cir. 1991) and *In re Keller*, 642 F.2d 413, 425, 208 USPQ 871, 881 (CCPA 1981).

"Section 103 forbids issuance of a patent when 'the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.'" *KSR Int'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1734, 82 USPQ2d 1385, 1391 (2007).

"The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results."

Leapfrog Enter., Inc. v. Fisher-Price, Inc., 485 F.3d 1157, 1161, 82 USPQ2d 1687, 1691 (Fed. Cir. 2007) (quoting *KSR Int'l v. Teleflex, Inc.*, 127 S. Ct. 1727, 1739, 82 USPQ2d 1385, 1395 (2007)). "One of the ways in which a patent's subject matter can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *KSR*, 127 S. Ct. at 1742, 82 USPQ2d at 1397.

ANALYSIS

Claim 1

The Examiner characterizes the library calls of Poulsen as the claimed second program unit that includes a set of one or more instructions to partition a parallel operation among a plurality of threads and the reference to the privatizable storage object declarations as the third program unit (Answer 17). The Examiner specifically argues that the library calls 260 and 270, which comprise operations that are partitioned among multiple threads, are actually instructions to initialize parallel regions of the program and reference the privatizable storage object declarations in block 220 (*id.*).

Appellants assert that the library calls of Poulsen do not partition a reduction operation between threads, nor do they reference a third program unit which is also translated from the first program unit (Br. 12-13). According to Appellants, the privatizable storage object declarations disclosed in Poulsen are not an encapsulation of a reduction operation (Br. 13). Appellants further argue the combinability of the references by stating

that Sundaresan does not discuss program translation and therefore, is not relevant to Poulsen (*id.*).

We disagree with Appellants that the library call of Poulsen does not partition the operation between a plurality of threads since the parallel regions actually instantiate and initialize the thread-private memory used in parallel processing (FF 4 & 8). Additionally, we remain unconvinced by Appellants' assertion that the privatizable storage object declarations do not encapsulate a reduction operation. In that regard, as pointed out by the Examiner (*supra*), block 220 declares a privatizable storage object which encapsulates the current global storage object (FF 7) to be accessed by multiple processors or threads (FF 2-4).

Additionally, contrary to Appellants' assertion against combining the references (Reply Br. 3), combining Sundaresan with Poulsen does not require a teaching of both partitioning parallel operations as well as using a reduction operation and a parallel operation in both references. The parallel operation disclosed by Sundaresan is a data-parallel reduction operation performed by a group of threads (FF 10). In fact, since Sundaresan's reduction operation may be reused for different operation types (FF 9-11), one of ordinary skill in the art would have combined the references to perform the reduction operation using the parallel computer programming of Poulsen to benefit from the parallelism-enhancing effects of privatization of global storage object (FF 2 & 3).

Claims 23 and 26

Appellants present similar arguments for these claims and argue that Poulsen merely addresses libraries and the privatization of storage objects,

instead of the claimed using a run-time library or a library to perform a reduction operation (Br. 14). We disagree. As discussed above and argued by the Examiner (Answer 20-21), the rejection is based on using the reduction operation or the algebraic operation of Sundaresan as the operation to be parallel processed in Poulsen (FF 9).

Claims 22, 24, and 25

Appellants argue that Hardwick does not teach performing a plurality of "vector operations," as recited in claim 24 (Br. 14) and may not be combined with Poulsen and Sundaresan since Hardwick is directed to reducing inter-processor communications during parallel processing, instead of being concerned with program translation (Br. 15). The Examiner responds that there is no need for multiple vector operations to be disclosed by all the references since Hardwick's vector operation may be used in each of the plurality of reduction operations of Sundaresan (Answer 22).

We agree with the Examiner and find that Hardwick does indeed describe a vector operation to be carried out over a group of processors operating in parallel (FF 12 & 13) such that each of the reduction operations of Sundaresan may benefit from implementing such vector operation. Not only does Hardwick discuss the benefit of portability, but also one of ordinary skill in the art would have performed program translations to implement the vector operation as Hardwick does provide for translation of the vector operation for parallel processing (FF 13).

CONCLUSION OF LAW

Because Appellants have failed to point to any error in the Examiner's position, we sustain the § 103 rejection with respect to claims 1 and 22-26, and also with respect to claims 3, 6-8, 10-15, 18, 20, and 21, which are argued together with claim 1 (Br. 14). Therefore, we sustain the 35 U.S.C. § 103 rejection of claims 1, 3, 6-8, 10-15, 18, 20, 21, 23, and 26 over Poulsen and Sundaresan, and of claims 22, 24, and 25 over Poulsen, Sundaresan, and Hardwick.

DECISION

The decision of the Examiner rejecting claims 1, 3, 6-8, 10-15, 18, and 20-26 is affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

KIS

TROP, PRUNER & HU, P. C.
1616 S. VOSS ROAD, SUITE 750
HOUSTON, TX 77057-2631